Free University of Bozen-Bolzano

Faculty of Computer Science

Mobile Systems Engineering
ANDROID APPLICATION PROJECT
FINAL REPORT

Prepared by:
Vytautas Jankauskas

Supervisor:
Prof. Nabil El Ioini

2015/2016

Abstract.

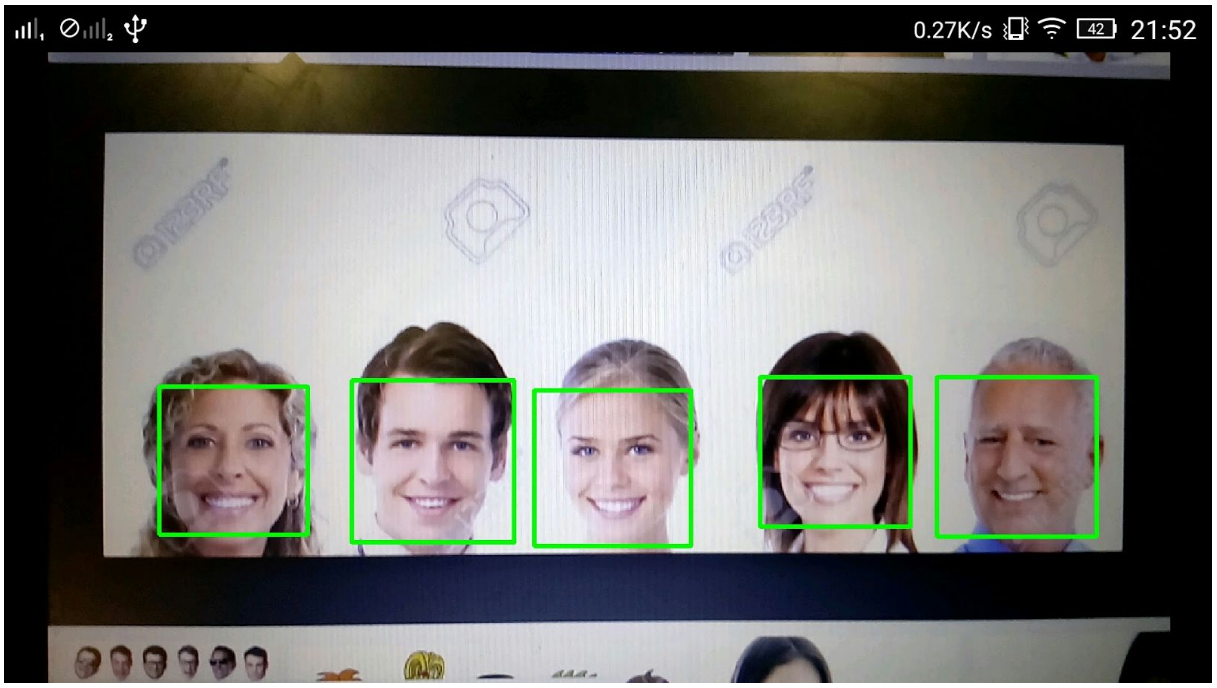
The main goal of this project is to develop an Android application which uses camera of a smartphone to detect human faces on a real time and to output a number of faces detected to the Arduino microcontroller.

Idea.

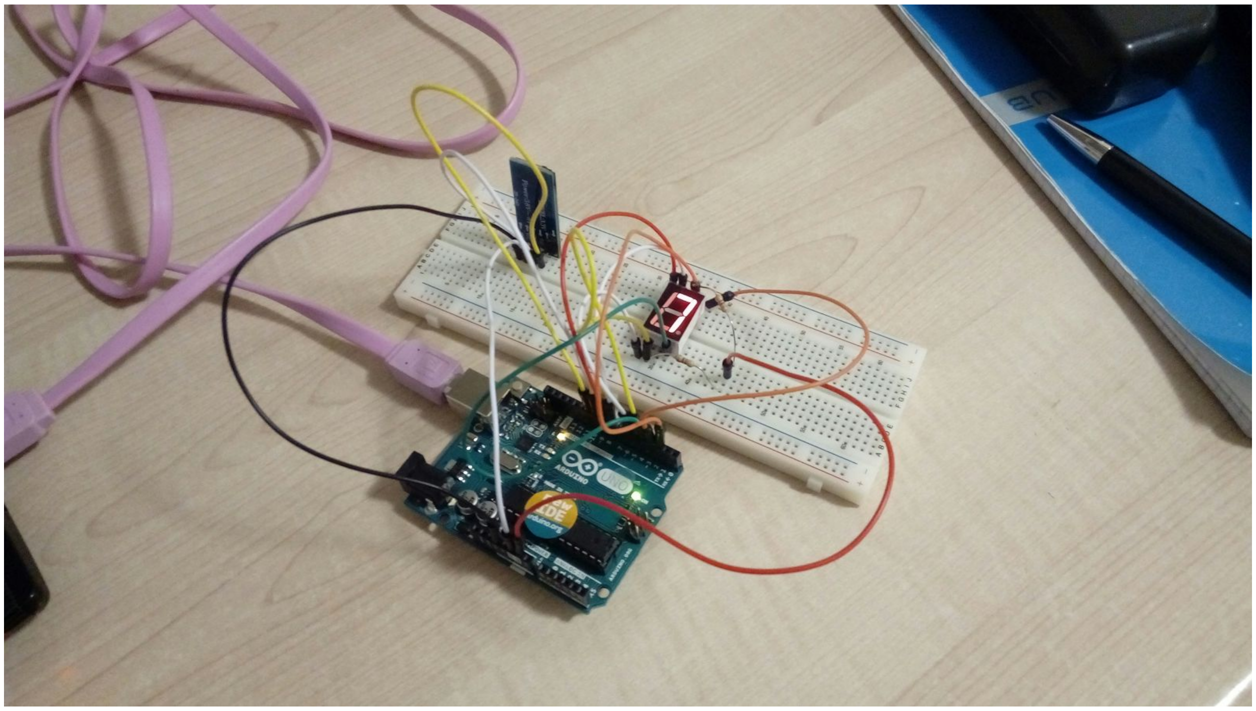
To apply the algorithm of face detection from OpenCV computer vision library in an Android application. The application connects to an Arduino microcontroller Bluetooth module and then broadcasts a number of real time detected faces to a 7 segment LED display.

Features.

- User is capturing the camera view and sees the detected faces in the screen of a smartphone (every face detected is represented as a rectangle with green borders).



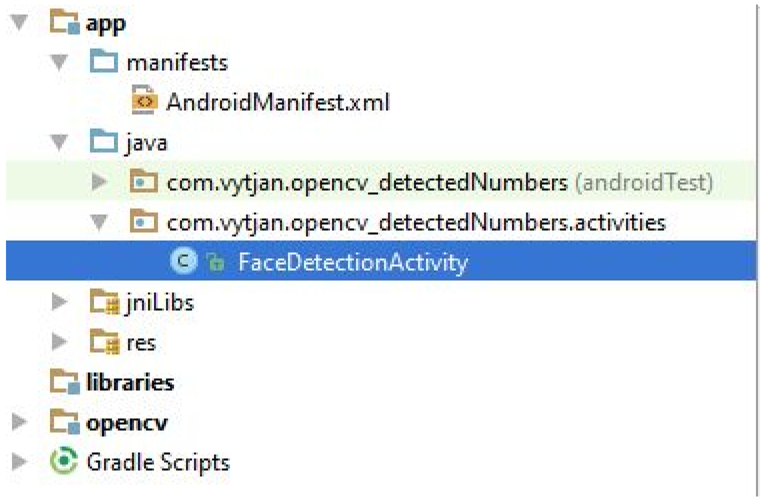
- If the user smartphone has a Bluetooth connection available and an Arduino with Bluetooth model and 7 segment LED screen is connected, then a number of faces detected is displayed in it.



Required specification of Android device: Android API Level 10 or higher, Camera, Bluetooth (optional for full functionality).

OpenCV library.
OpenCV (Open Source Computer Vision Library) is used as a base for face detection algorithm in this project. It is an open source computer vision and machine learning software library (http://opencv.org/about.html).

Architecture.



The application has one main class - *FaceDetectionActivity*. *FaceDetectionActivity* class contains all the application logic (excluding used methods of the OpenCV library, which are implemented in files in *jniLibs* and *opencv* directories). The execution flow of this application (in *onCreate* method) is:

1. Check if Bluetooth connection is available, if yes, then try connect to Arduino Bluetooth module.
2. Bind an instance of OpenCV library *JavaCameraView* to the application layout.
3. Set Cv*CameraViewListener* to start face detection process.

Other main methods (*onPause, onResume, onDestroy, onCameraViewStarted, onCameraViewStopped*) are used to handle necessary steps to close/resume/pause application without crashes.

Besides default Android Activity methods, this class contains all the logic needed for successful implementation of this application:

1. *Face detection*.
    a. To achieve the wanted result, we are using a *CameraBridgeViewBase*. It implements an interaction with Camera and OpenCV library. According to the documentation (http://docs.opencv.org/java/2.4.8/org/opencv/android/CameraBridgeViewBase.html), the main responsibility of it is to control when camera can be enabled, process the frame, call external listener to make any adjustments to the frame and then draw the resulting frame to the screen. The clients shall implement CvCameraViewListener (*FaceDetectionActivity implements a CvCameraViewListener2*).
    b. Method *BaseLoaderCallback* is used to load an OpenCV library and cascade file *lbpcascade_frontalface.xml.* Cascade file defines a front-face (which is needed to be detected) layout in terms of the dots, e.g. there are

   dots defined for the pupil, right-cheek, left temple etc. Cascade file used was trained on 3000 positive face samples and on 1500 negative face samples.
   Then *mJavaDetector* is initialized. It takes as an argument the same cascade file and proceeds all the calculations needed to detect a pattern of face described in cascade file. It is based on Local Binary Pattern (LBP) vector classification type (more about the LBP: http://www.scholarpedia.org/article/Local_Binary_Patterns).
   c. Method *onCameraFrame* is used to process the output data of every captured frame (in other words, to draw green rectangles around faces detected and to push a number of faces detected to a Bluetooth socket which broadcasts that number to Arduino 7 segment LED screen).

2. *Bluetooth connection with Arduino.*
   a. *Method CheckBt* is used to check if Bluetooth is presented in a current smartphone and if it is turned on.
   b. Method *bluetoothConnection* establishes Bluetooth connection with Arduino Bluetooth module using given Bluetooth device adress and UUID.
   c. Method *beginListenForData* creates instance of a Bluetooth socket and listens for data on it and transfers it to an Arduino board.

Arduino.
Arduino Uno microcontroller was used for this project because it is relatively easy to use, it is open source, it has a large community of users and developers. The 7 segment LED screen was selected to show a number of faces detected because it is relatively easy to set it up on Arduino and the “showing a number” algorithm is simple to implement. Since it has 7 segments, to turn on every one of them, a separate pin on Arduino board is used. A *DisplayDigit* method in code, uploaded to Arduino is responsible for setting up what segment to turn on for different numbers.

Difficulties, problems.
During the development of this application, few major problems were faced.
1. It is possible to implement a face detection algorithm in 2 different ways - using native (C++) code of an OpenCV library or using a JAVA interface of OpenCV library. However, during the development I did not manage to successfully set up Android NDK and make native code work on the application. To solve this problem, usage of JAVA interface was selected.
2. Establishing a Bluetooth connection with an Arduino Bluetooth module was another tricky part. Default Arduino Serial library used for communication

between Arduino board and a smartphone did not work at all. To solve this problem, a research was done and a SoftwareSerial library for communication was selected. This library allows to communicate with Arduino not only with built - in serial communication on pins 0 and 1, but it also allows serial communication on other digital pins of the Arduino, using software to replicate the functionality.

3. The face detection algorithm itself is still not very accurate as sometimes it does not detect a human face, sometimes it detects non-existing human face.

Future plans.

Future plans of this project are to use it as a prototype of a future application in a smart home system technology, or tracking of a number of people coming to the supermarket/football stadium/etc. Another step to develop and implement is a face recognition. By doing that, the use cases of the application could be integration to security cameras or setting up a new camera outside the entrance to the house and detect a number of people visiting or even to recognize/identify those people and act accordingly (e.g. voice message defining identity of people visiting, turning on the lights, music, closing/opening of curtains etc.).
Another thing for the future is to improve the accuracy of the face detection algorithm.